6. query() method

This method allows to query data from a DataFrame using Boolean expression in a more readable and concise way compare to Boolean masking.

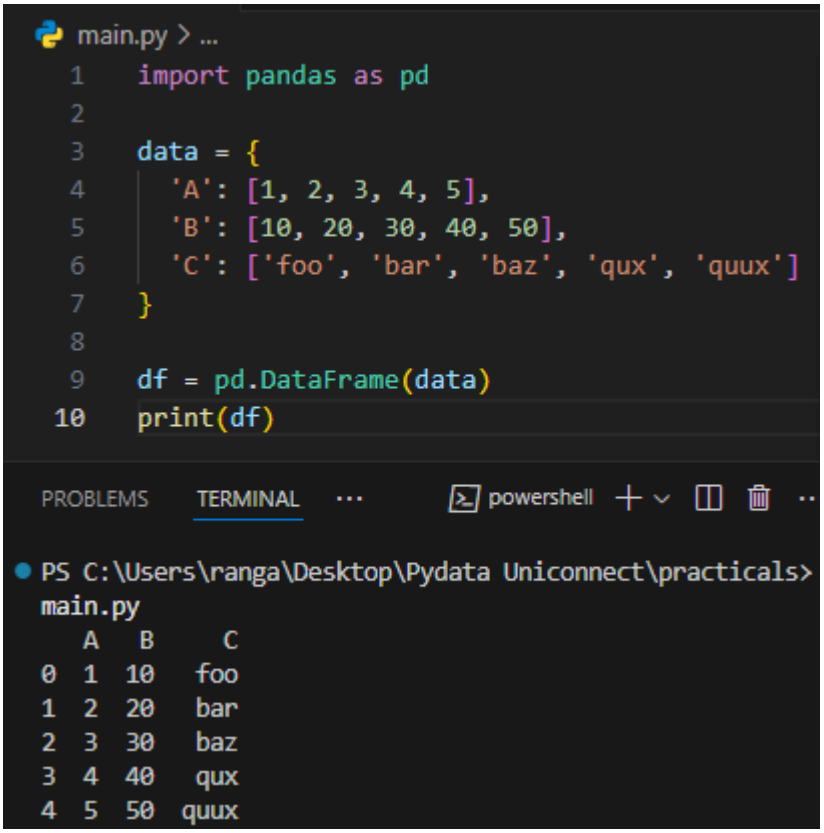
Syntax,

```
DataFrameName.query(expression, inplace=False, **kwargs)
```

- `expression` : The query string.(Should be a valid Boolean expression involving columns of the DataFrame)
- `inplace` : If `True` , filtering will be done in-place and the original DataFrame will be modified.
- `kwargs `**: Additional keyword arguments passed to the underlying` eval` function.

This method returns a new DataFrame with the rows that match the Boolean expression.

Ex:



1. Querying a single column

```
11  result = df.query('A > 2')
12  print(result)
```

```
PROBLEMS  TERMINAL  ...  powershell
PS C:\Users\ranga\Desktop\Pydata Uniconnect\practicals>
   A   B    C
2  3  30  baz
3  4  40  qux
4  5  50  quux
```

2. Querying multiple columns

```
11  result = df.query('A > 2 and B > 30')
12  print(result)
```

```
PROBLEMS  TERMINAL  ...  powershell
PS C:\Users\ranga\Desktop\Pydata Uniconnect\practicals>
   A   B    C
3  4  40  qux
4  5  50  quux
```

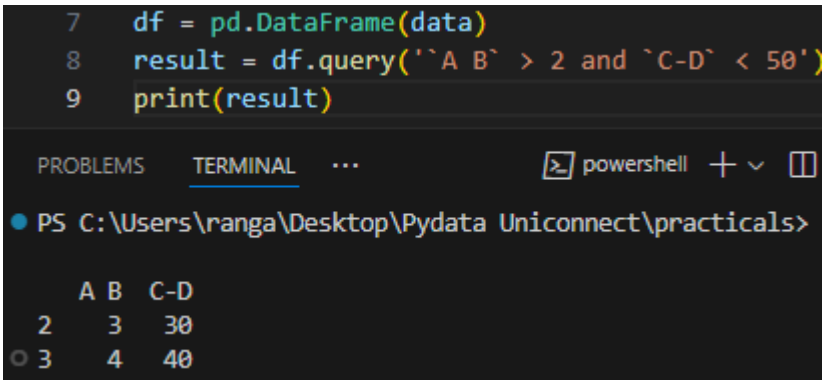
We can use Local Variable within our query string.
Ex:

```
11  threshhold = 5
12  result = df.query('A < @threshhold')
13  print(result)
```

```
PROBLEMS  TERMINAL  ...  powershell
PS C:\Users\ranga\Desktop\Pydata Uniconnect\practicals>
   A   B    C
0  1  10  foo
1  2  20  bar
2  3  30  baz
3  4  40  qux
```

Colum names with spaces or special characters can still be used but need to be enclosed in backticks. (``)
Ex:

```
data = {
        'A B': [1, 2, 3, 4, 5],
        'C-D': [10, 20, 30, 40, 50]
}
```



Additional :-

`eval` functions

The `eval` function in pandas is used to evaluate string expressions in the context of the calling DataFrame.
When you use the `query` method, pandas internally utilize the `eval` function to parse and execute the query string.

Ex:

```
data = {
        'A': [1, 2, 3, 4, 5],
        'B': [10, 20, 30, 40, 50],
        'C': [100, 200, 300, 400, 500]
}
df = pd.DataFrame(data)
result = df.eval('D = A + B + C')
print(result)
```

	A	B	C	D
0	1	10	100	111
1	2	20	200	222
2	3	30	300	333
3	4	40	400	444
4	5	50	500	555

Advantage of `eval`

- `eval` can be faster than python's regular standard operators because it uses `numexpr` for efficient computation, especially with large DataFrames.
- It allows for concise and readable expressions without the need for looping or multiple lines of code.

- Ability to handle complex expressions involving multiple columns and operations.
  Warning
  Since `eval` executes the string as code, ensure that the input strings are trusted and controlled to avoid code injection vulnerabilities.